Int. J. Human–Computer Studies (1998) **48,** 777–801

The design and evolution of TurboTurtle, a collaborative microworld for exploring Newtonian physics

Andy Cockburn
Department of Computer Science, University of Canterbury, Christchurch, New Zealand
email: andy@cosc.canterbury.ac.nz

Saul Greenberg
Department of Computer Science, University of Calgary, Calgary, Canada
email: saul@cpsc.ucalgary.ca

(Received 22 January 1998)

TurboTurtle is a dynamic *multi-user* microworld for the exploration of Newtonian physics. With TurboTurtle, students can alter the attributes of the simulation environment, such as gravity, friction, and presence or absence of walls. Students explore the microworld by manipulating a variety of parameters, and learn concepts by studying the behaviours and interactions that occur. TurboTurtle has evolved into a "group-aware" system where several students, each on their own computer, can simultaneously control the microworld and gesture around the shared display. TurboTurtle's design rationale includes concepts such as equal opportunity controls, simulation timing, concrete vs. abstract controls, recoverability, and how strictly views should be shared between students. Teachers can also add structure to the group's activities by setting the simulation environment to an interesting state, which includes a set of problems and questions. Observations of pairs of young children using TurboTurtle highlight extremes in collaboration styles, from conflict to smooth interaction. Finally, the technical work in making TurboTurtle group-aware is slight, primarily because it was built with a groupware toolkit called GroupKit.

1. Introduction

The 1990s and the new millenium will see the Internet pervade through the educational system. Approximately half of the US public schools are already connected to the Internet (Press, 1996), and many politicians advocate Internet access in every classroom (for example, US President Bill Clinton's 1996 State of the Union Address). For many people the Internet is synonymous with the World Wide Web. However, the Internet also provides a network infrastructure for real-time communication that enables a variety of novel computer applications for supporting collaboration. These applications include audio and video conferencing, shared electronic white-boards, collaborative text editors and so on. Commercial vendors are rapidly developing applications that exploit the real-time communication capabilities of the Internet and examples include Intel's "ProShare",† Netscape's "CoolTalk"‡ and MicroSoft's "NetMeeting".§ While these

†http://www.intel.com./comm-net/proshare/.
‡http://cgi.netscape.com/eng/mozilla/3.0/relnotes/cooltalk/.
§http://www.microsoft.com/netmeeting/.

1071-5819/98/060777 + 25$25.00/0/hc970179

preliminary computer supported collaboration tools evolve, we expect educators to consider how the Internet can support collaboration between students in real-time.

We are investigating the application of real-time collaboration-aware software systems within educational environments. Our background is that of researchers in computer-supported cooperative work (CSCW), an interdisciplinary research field which attempts to understand cooperative activities and to mould computer technology so that it supports and enhances those cooperative activities (Baecker, 1993). The software systems resulting from CSCW research are termed "groupware". Although much has been learned about how *adults* work together through groupware, little attention has been paid to how children collaborate through real-time groupware.

In this paper, we discuss the critical issues of design and usability within the context of the construction and use of an educational collaborative artifact. The purpose of the artifact is to provide a focal-point of discussion and interaction for children. In our case, this artifact is a Newtonian microworld called TurboTurtle, and the students could be children as young as 7 years and as old as 17 years. Critical issues in the design and motivation of TurboTurtle include the following.

1. Much of our design rationale in developing the TurboTurtle user interface is directed at producing an educational environment that is both engaging and easy to use. Students will only explore the microworld if it is engaging (Soloway & Pryor, 1996).
2. As a free-form microworld, students can manipulate TurboTurtle as they wish. However, teachers can tailor TurboTurtle to display a prescriptive set of tasks containing questions, lines of investigation and hints of things to try. We believe that this allows teachers to scaffold the student's passage through TurboTurtle's educational domain.
3. TurboTurtle is a truly collaborative microworld, where students have their own displays, their own mouse and an ability to do anything at any time. We believe that communication around the microworld is reinforced by explicitly providing support for simultaneous collaborative activities such as mutually setting simulation properties, gesturing around the display and pointing to microworld objects. Since students do not have to be co-located, we assume that they can talk to each other over an audio channel such as a speaker-telephone. It provides a platform for investigating styles of collaboration beyond shared use of single-user systems.

We intend that TurboTurtle should be pedagogically worthwhile within its Newtonian domain, but to date our primary interests in this research project lie in the design and usability issues of computer support for real-time collaboration for children.

We present the design of TurboTurtle, and how it evolved into a collaborative microworld. The paper begins by briefly reviewing microworlds and educational issues. Section 3 introduces the single-user version of TurboTurtle and its facilities for tailoring its educational support to the teacher's curriculum, and Section 4 describes the final collaborative version. Each section describes what the system looks like to the student, and highlights the design rationale for the features we believe critical to make TurboTurtle a useful educational system. Section 5 describes our preliminary observation-based evaluation of collaborative TurboTurtle with pairs of primary school children. Section 6 reviews and contrasts related work on collaborative microworlds. The paper

closes by examining the technical effect involved in making the microworld group-aware through a groupware toolkit called GroupKit (Roseman & Greenberg, 1992, 1996).

2. Background: microworlds

Microworlds, or computer simulations of restricted environments, are an intuitively appealing way to promote discovery and exploratory learning (Smith, Cypher & Spehrer, 1994). One type of microworld, and the subject of this paper, simulates an adjustable Newtonian universe. In it, students can experiment with concepts such as gravity, friction, force, velocity and so on, and see how changes in their value affect the objects moving within the simulation.

Microworlds—Newtonian or otherwise—are not new. They were first conceptualized by Papert in his 1980 book "Mindstorms", but in that era they were implemented as crude systems that required students to adjust the simulation via cryptic and error-prone command line interfaces, e.g. Logo. The wide-spread introduction of graphical interfaces in the late 1980s and the early 1990s then changed the way educators presented microworlds to students. The simulations became dynamic environments that students could alter on the fly, usually by changing property settings on control panels and by directly manipulating the objects within the world. The Alternative Reality Kit, an intriguing Newtonian microworld built in 1987, is one such example (Smith, 1987). In this paper, we claim that another evolutionary step is about to take place: microworlds will become *group-aware* by actively allowing several students to view and manipulate the simulation.

Papert called computer-supported microworlds "incubators for knowledge" when he described the potential of computer-aided learning to encourage exploration and thus self-education by children (Papert, 1980). His educational philosophies stem from Piaget's work on learning† which, simplistically, state that much of children's learning occurs without being taught: children construct their skills and understanding from seeds of knowledge. Many accepted Papert's beliefs on the educational value of exploratory learning as established fact (Maddux, 1985).

Yet not all educators agree with Papert. In sharp contrast to microworlds, many educational systems are highly prescriptive. They direct students through small increments of information, and test whether the student has mastered the material before advancing. The advantage derived from such directed learning is that teachers make the learning goals explicit, and that students can advance at their own pace. Exploratory systems such as microworlds, on the other hand, have little or no motion of predetermined trails that students must follow. Students are equally free to test personally derived hypotheses, or meander (some would say mindlessly) through the uncontrolled environment.

Rather than strongly advocate either exploratory learning or perspective learning, as software developers we believe that software systems should provide tailorable scaffolding support which allows teachers, mentors or parents to configure and adapt the system

† A review of Piaget's work, or that of any educational psychologist is beyond the scope of this paper. Interested readers should consult Gruber and Voneche (1987) or Papert (1993) for further reference.

to different students and learning styles (Soloway & Pryor, 1996; Rossen & Carroll, 1996). TurboTurtle's scaffolding support is described in Section 3.4.

Aside from issues of appropriate scaffolding, microworld proponents claim that a major part of their educational value comes from providing an artifact around which children can discuss their work (Maddux & LaMont-Johnson, 1988). In current practice, this discussion and sharing is realized by having two or more students view and manipulate a "single-user" microworld. They collaborate by sharing access to the computer's input devices (keyboard, mouse) and its output (the screen)—related work with alternative sharing mechanisms is discussed in Section 6. The fact that there is only one set of input controls means that only one student can be "driving" the simulation at a time. While control of the mouse is not a reliable indicator of who is controlling the group collaboration (Cole, 1995), it does often cast the group members into roles that may be hard to relinquish, or that have certain social overtones. For example, the role of a mouse driver can vary between passive scribe (Mantei, 1988; Cole, 1995) to decision maker (Klawe & Phillips, 1995). People who are not controlling the mouse may feel a loss of power, even when the driver is obeying their directions (Cole, 1995). Because only one person at a time can use the single mouse, it becomes an introduced artifact that can alter the group's dynamics in unpredictable ways. TurboTurtle's support for collaboration (Section 4), in which user has their own display and input devices, allows us to investigate childrens' collaborations around a shared work surface without these constraints on shared hardware devices.

The following two sections detail the design of the TurboTurtle microworld. Its usability as a single user system including support for scaffolding are described, followed by a description of the evolution of its support for real-time collaboration.

3. The solitary microworld

This section considers the usability of a solitary microworld designed for one student, concentrating on the design rationale we used to construct the student's interface. Since many of our choices were motivated by the look-and-feel of contemporary microworlds, the interface style follows the current genre. What is important is that we highlight what we believe are the more unusual but still critical factors that affect the system usability.

3.1. TURBOTURTLE-1

Our work with microworlds began in 1986 (described in Cockburn, 1994). TurboTurtle-1 supported a full interpreted dialect of Logo with standard list processing, program control-flow constructs and turtle-graphics. We transformed Logo into a Newtonian microworld by providing a new set of "dynamic turtle graphic primitives". Through a command-line interface, students controlled various Newtonian properties of the world, such as gravity, friction and the presence or absence of walls. For example, they could turn off gravity by typing `Gravity 0`. Students could also manipulate the "turtle" (a movable ball) by adjusting its position, velocity and mass; changing its kinetic and potential energy; and applying a force to it by strapping a rocket to its back. For example, `Rocket 50 10` would propel the turtle with a force of 50 units for 10 time units. Students then viewed the effects of their commands on the simulation, where the turtle

would be seen moving across the display according to the current set of Newtonian values. Unfortunately, hardware limitations prohibited students from dynamically altering an on-going simulation. They had to stop the animation, reset the values and restart.

The capabilities of the system were encouraging, but the command line interface lacked the potential of engaging the users. Students had to learn and remember the syntax and commands of the language. They had to type accurately, a far from trivial task for young children. Changing simulation parameters was slow, and viewing the results was incremental.

3.2. TURBOTURTLE-2

Our second iteration, TurboTurtle-2 (Cockburn, 1994) moved away from Logo and the command-line interface towards a fully graphical system. We wanted to provide a seamless interface that allowed all the student's cognitive effort to be directed at the contents of the microworld. Beyond the "see and point" premise of modern graphical user interfaces (Shneiderman, 1987), we wanted TurboTurtle-2 to make extensive use of sound, colour and animation to capture the interest of young users.

What do students using TurboTurtle-2 see and do? Figure 1 is a snapshot of a student's session. The lower-half displays the simulation with the turtle being the ball at its centre. The turtle's location can be changed directly by dragging it with the mouse,

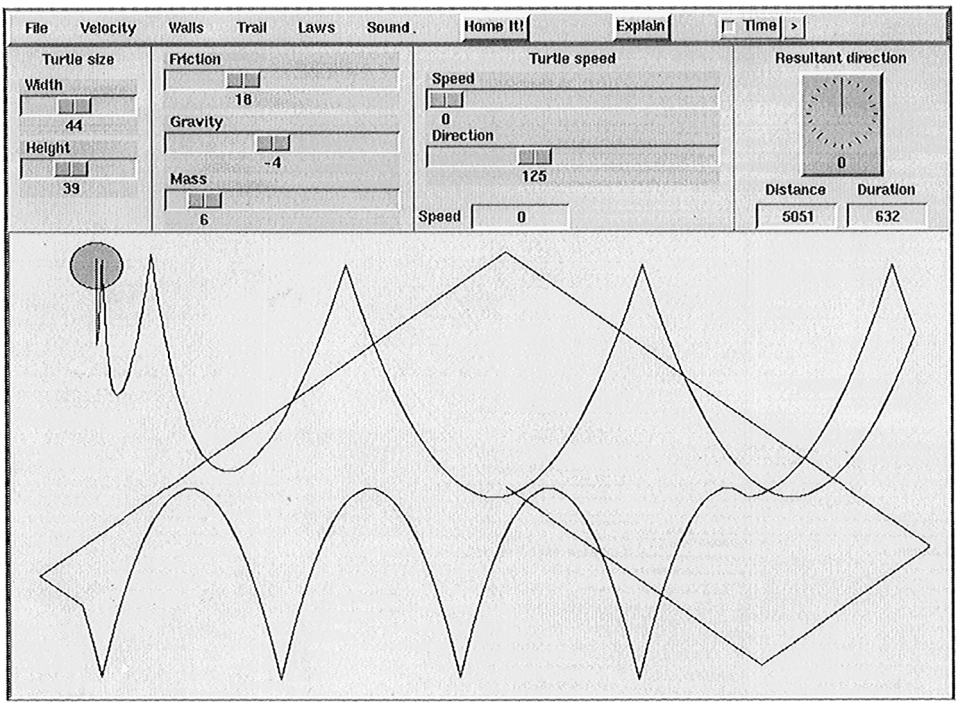

FIGURE 1. The main window to TurboTurtle2.

and its direction and velocity altered by "throwing" it. The top of the figure shows a control panel, where tangible properties are set through constantly visible graphical sliders. These include the controls to change the turtle's size, mass, speed, the degree of friction and gravity and so on. Students use the pull-down menus to access advanced features of TurboTurtle-2.

Within the simulation, the turtle's trail, a line of ink that follows the turtle's movement, can be switched on or off. The walls in the microworld can be changed as well. The turtle bounces off "hard" walls and passes through "transparent" ones (which causes it to wrap-around the display). When only the ground is hard, the relative location of the ground to the turtle is remembered as it wraps through successive screens. Students can also display a mountain scenery backdrop (visible in Figure 3), which provides additional visual cues to the altitude of the turtle. As the turtle gains altitude the backdrop changes to show smaller mountains, a row of aeroplanes and then satellites. Of course, the trails and the mountain backdrop can be cleared at any point.

Figure 1 shows the turtle's trail after a series of user-driven changes to the microworld.† Starting in the middle of the screen, the turtle moved down and to the right with no mass or gravity. After seeing and hearing it bounce off the walls four times, the student added mass and positive gravity, causing the turtle to bounce under gravity (the sin curve). She then changed gravity to a small negative value, causing the turtle to bounce off the roof of the microworld. Finally, she added friction, causing the turtle to eventually slow to a stop.

3.3. DESIGN DECISIONS

Several of the design decisions that are part of TurboTurtle-2 are not immediately obvious from the interface description. These are presented as a set of concepts that we believe are applicable to microworld simulations.

Equal opportunity controls incorporate the idea that interface controls can serve as both input and output (Thimbleby, 1990). This reduces interface clutter by avoiding duplication of screen components for control and feedback. For example, TurboTurtle-2 includes equal-opportunity sliders. Students can (say) set the speed and direction of the turtle by using the two sliders shown in the top-middle of Figure 1. But Newtonian effects such as gravity will cause these values to change as the simulation is running. Instead of keeping the slider static, it will automatically update and animate itself to reflect the instantaneous values of the property it is monitoring.

As well as being a good interface approach, equal opportunity controls have educational value. First, students now have a quantitative measure of what the turtle is doing at any point in time. Second, individual properties of the simulation that are perceptually hard to view can be teased out. For example, the turtle's rate of movement can be displayed as its vector components of speed and direction. Finally, the movement of the slider thumb gives additional information on the turtle's activity by animating the

† Naturally, the figure fails to show the turtle's movement, the dynamically changing slider values, the colour and the audio output that are fundamental to the user's sense of engagement.

monitored value: for instance, the rate of change of velocity is vividly revealed by slider movement, but is hard to visualize in a textual display.

While sliders represent how input controls can show output, so can output act as input. The turtle itself is an equal opportunity control. As mentioned earlier, students can grab the turtle and drag it to a new position causing corresponding changes to its potential energy. When they 'throw' it with the mouse, it starts moving in the given direction at the speed it was thrown. Of course, all these changes are continuously reflected by the sliders.

Timing. In TurboTurtle, the microworld clock (set by the *time* button) lets the student freeze the microworld at any point. This allows specific values to be set prior to running a new experiment. Time can run smoothly, giving a continuous real-time simulation, or discretely which allows students to scrutinize the change in variable values at critical instants. For example, the student could investigate changes in potential and kinetic energy by discretely stepping through the turtle's motion as it hits the floor and as it reaches the apex of its motion under gravity.

The user controlled clock also allows the user to pause the dynamic behaviour of the equal opportunity interface controls. Without it, our extensive use of equal opportunity *could* make parts of the simulation hard to control. With the clock running, the student's attempts to set a specific value would be constantly affected by the system's dynamic modifications to the same set of controls.

A design limitation in the current version is that time can only run forwards. The ability to run time backwards is pending implementation.

Concrete and abstracted controls. TurboTurtle is intended for students ranging from 7 to 17 years in age, and for peer groups where individuals have different knowledge and talents. How can this divergent audience be handled in a microworld?

Our solution was to create two sets of controls: concrete and abstract. Concrete controls, which are continuously visible, present concepts that are familiar and frequently accessed by the youngest students (as shown in Figure 1). Abstract controls for more sophisticated manipulations are revealed on demand by mature users. For example, TurboTurtle-2 lets advanced users view and manipulate values in Kinematic equations, which are selected as menu options in the "Laws" pulldown menu (Figure 2, left-side). Choosing the first "Energy" equation creates a window into the microworld that dynamically displays the turtle's potential and kinetic energy. The second "Rocket" equation creates a control panel (Figure 2, right-side) that allows students to attach rockets of varying force and fuel times to the turtle, which lets them examine the inter-relation between force, acceleration, mass, gravity and friction. Other kinematic equation options provide dynamic simulations of the behaviour of a user-specified set of formula values: essentially they provide an animated calculator.

Recoverability is an important property in any interface that encourages exploration. Even a seemingly benign microworld has elements of the risk for students. Recoverability allows users to experiment with features, safe in the knowledge that they can get back to their starting state.

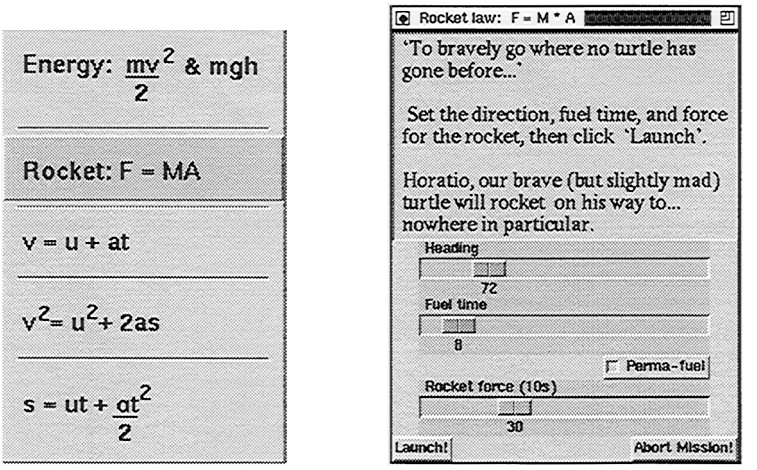

FIGURE 2. Selecting and using "formal" experiments in TurboTurtle.

Exploring a dynamic microworld is risky because it can change rapidly. In TurboTurtle-2, for instance, a student may arrange a group of slider values to simulate a rocket working against a certain friction, gravity and mass. When the rocket is launched, the simulation runs and slider values will change to reflect the dynamically changing environment. In early trials of the system, we would note that students frequently forgot or mistook one or more slider values. When they ran the simulations, they were often immediately aware of their error, and found it annoying to have to reset the values that the system had changed. Similarly, students may be reluctant to change system parameters away from an interesting state for fear of corrupting them.

TurboTurtle-2 lets students recover from their ventures by allowing them to save and reload named states of the microworld. Of course, this is an explicit action that students must take, and they will likely do this only for highly interesting states. The proposed time control extensions mentioned previously, whereby the simulation can be run backwards, is also a type of undo, allowing students to return to prior critical points.

3.4. SUPPORTING THE TEACHER'S CURRICULUM

As described so far, TurboTurtle is a free-form exploratory microworld that provides no guides to appropriate paths through the environment that it simulates. TurboTurtle therefore includes a scaffolding facility that lets teachers, mentors or parents provide students with directions that guide and assist exploration of the Newtonian world. Rossen and Carroll (1996) describe the roles of "scaffolding examples" as "sample problems of realistic size whose complexity is gradually revealed in steps that leverage and reinforce the intrinsic structure of the problem-solution process … Scaffolding examples exploit the natural dependencies among a domain's concepts, tasks and procedures".

The main idea of TurboTurtle's scaffolding facility is that an "interesting" state of a microworld simulation is used by the teacher to create a task. The teacher first uses TurboTurtle to run the simulation until it reaches a desired configuration. The teacher saves this state (through TurboTurtle's state saving facility) to create a "scenario" that specifies a set of values in the microworld. They can then annotate this scenario with directions, questions and hints.

The students start their explorations by calling up the scenario (which could be assigned to them by the teacher). Because the state of the microworld is established automatically, all students using this scenario start with the correct settings, and avoid the errors that occur if they had to recreate the settings themselves. As part of the scenario, the teacher's directions are displayed (bottom right window in Figure 3). Student's then carry out the actions described, and (optionally) type in their observations, conclusions or answers in another dialog box (not shown). Of course, students are not forced to follow the teacher's instructions. In keeping with the exploratory microworld philosophy, the teacher's guidance is a recommended track that people can experiment around, rather than follow strictly.

Figure 3 shows TurboTurtle immediately after a student has loaded a scenario. In this case, the teacher wants students to learn that objects with zero mass are not influenced by gravity. The scenario gives the turtle zero mass and the world no gravity, and the turtle is slowly moving across the screen leaving a trail in front of a mountain scene. Students continue their experiments from this point. They are first asked to form a hypothesis on what will happen when gravity is added. They are then directed to actually do this, and to

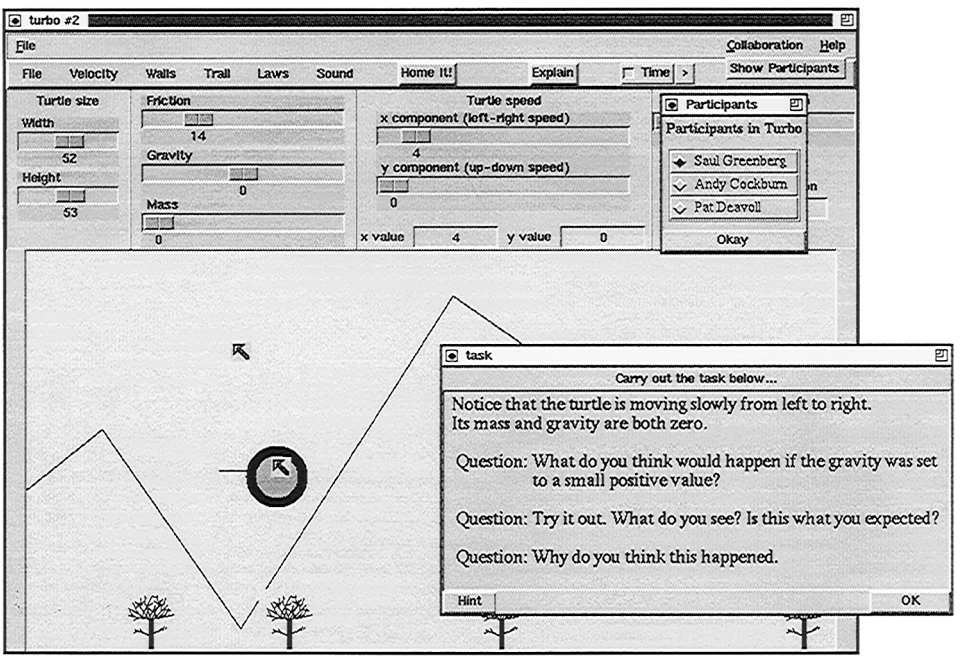

FIGURE 3. Setting a scenario and task.

comment on whether the results they see are the ones expected. The hint, when selected, will tell them to add mass to the object.

Teachers have full control of the contents of perspective tasks which lets them adapt TurboTurtle to particular teaching and learning styles. Teachers can vary the amount of explicit direction by deciding on the degree of freedom that a scenario should offer, by controlling the number of directions and hints provided, and so on.

4. The communal microworld

The next step was to redesign the microworld into a communal one, where small groups of students (diads or triads) could manipulate and talk about the simulation. We wanted to reuse much of the design of the solitary world for several reasons. From a learning view, we wanted the same system to be usable by a single student as well as a group of students. From a design view, we were interested in the interface design decisions necessary to convert TurboTurtle-2 into a multi-user system. From a technical view, we wanted to see how much implementation effort was required to make the existing system group-aware.

This section is primarily concerned with the multi-user interface design decisions that governed the development of TurboTurtle-3 (Cockburn & Greenberg, 1995). It begins with an overview of the system, and continues with the design decisions made.

4.1. TURBOTURTLE-3

In static images, such as the screen snapshots in this paper, TurboTurtle-3 appears to be almost identical to TurboTurtle-2. Its group awareness, however, makes it style of use significantly different. In the description below, we focus on these differences by assuming that two or three distance-separated students, each with their own computer, are looking at the screen and are talking to each other by a speaker-phone.

Each student sees exactly the same running simulation on their display. The turtles are in the same position and move at the same speed, the trails are in the same place and the background scenery is identical. Similarly, the controls are *mostly* identical. They are in the same window location and have the same setting. However, students can decide to change their view of some of the controls. For instance, one could be examining turtle speed by its x–y components, and the other by speed and direction (Turtle meets Pythagoras!). Similarly, one could display independently some of the advanced control panels, such as the Energy panel.

In TurboTurtle-3 all students can work simultaneously doing anything they want at any time. For example, one student might move the turtle, while another adjusts its speed, and another alters the world's gravity. As in real life, they could even try to adjust the same control, which would cause it to "bounce" back and forth as they fight over its position! As any control is being adjusted, the new position is immediately reflected on all displays.

Only two new interface components were added to TurboTurtle-3. First, students see the other person's location on the screen by a *telepointer*, shown as the multiple cursors in Figure 3. Not only is a student's own cursor continuously drawn and updated on the display, but so are the cursor's of their partners. Next, a special menu option called

"collaborations" is added to TurboTurtle-3. This presents a dynamic list of all the students in the learning session (Figure 3, top right). Pressing a student's name will raise an information window describing that student.

4.2. DESIGN DECISIONS

In spite of the conceptual simplicity of interface changes, many design decisions had to be made. These included how students viewed the simulation, how they would control it, and how they could share their deictic references.

Viewing the simulation. What does it mean to have several students view the simulation? We considered four alternative approaches to view sharing.

1. *Strict WYSIWIS views.*† Every student would view exactly the same thing on their display: the ball as it was bouncing; the changes in background scenery; the ball's location in the scene; the tracing of ball movements; and so on.
2. *Relaxed WYSIWIS views.* While the state of the simulation would be the same, every student could have different viewports on it. That is, one student could be looking at (say) a zoomed out view, while the other could be zoomed in on a particular scene.
3. *Unconnected views, same simulation parameters.* The parameters of the simulation would be the same across all systems, but the effects of the parameters on the ball would be local. This could simply be a matter of each student's computer moving the ball at its own speed, but since performance of the computers would differ slightly, so would the position of the ball. Alternatively, a student could create a smaller simulation room by shrinking the window, which means that the ball would be bouncing off the walls at different places and frequencies. In either case, the ball position in the simulation would differ across the views.
4. *Unconnected views, different simulation parameters.* The parameters of each student's simulation would differ, thus affecting not only the position of the ball on a local display, but its overall behaviour as well.

We wanted the view to act as a conversational prop providing a focus for the students discussion (Hill, Brinck, Patterson, Rohall & Wilner, 1993). We therefore thought the strict WYSIWIS view would be the best choice to encourage this. The display becomes a shared cognitive artifact, and speech references would remain within the context of the shared image. Strict-WYSIWIS would allow students to pose questions and comments to each other such as "why did the ball bounce that way?" or "the ball just moved into outer space" or "look at the shape of the trace".

In contrast, views 2 through 4 would cause progressively greater breakdown in the discussion, probably resulting in greater confusion and ultimately less interaction between students (a similar observation was made by Tatar, Poster & Bobrow, 1991). Relaxed WYSIWIS causes people to ask "can you see this?" or respond "which one?"

† What-you-see-is-what-I-see, or WYSIWIS was coined by Stefik, Bobrow, Foster, Lanning & Tatar (1987) in a discussion about a shared whiteboard system.

Students using the unconnected view with the same parameters would have to explain what their ball is doing on their display. With different parameters, they would also have to explain the settings.

Although the relaxed and unconnected approach does give the student the ability to customize their view, the strict WYSIWIS view seems preferable as it reinforces the microworld's role as a conversational prop.

Controlling the simulation. The simulation is directed by manipulating the controls on the control panel: sliders, buttons, menu selections and by directly moving the ball position in the view. Given a strict WYSIWIS view and identical simulation parameters across the system, there remains several options for presenting the controls and for having students interact with them.

First, how do students view the controls? Controls could be identical on the displays (strict WYSIWIS), or different students may see different controls in their view (relaxed WYSIWIS). The choice is not clear here. In a complex simulation system such as TurboTurtle, the number of controls, including the pull-down menus and the pop-up panels, are huge and can clutter the display quickly. It seems reasonable to have a strict WYSIWIS view of the primary controls, while having a relaxed WYSIWIS view of advanced controls.

Second, how do students see the setting of a changed control? In a "parcel post" model (Tatar *et al*., 1991), the changed value of the control would be delayed until the student had completed their action. For example, if one student adjusted the gravity slider from 0 to 20, the other student would only see the slider jump instantly to 20. In contrast, the "interactive" model causes the control's state to be transmitted as it is being manipulated. Sliders move, buttons get pressed, pulldowns selected. Clearly, the interactive model is preferable, as students will be able to see the changes as they are made, and are less likely to miss the actions of the others.

Finally, who has permission to use what controls? Several choices are possible. Students could be assigned to a mutually exclusive subset of controls. Alternatively, a turn-taking model could be enforced, where only one student at a time can manipulate the controls. Or students may be allowed simultaneous access to all controls, constrained perhaps by some mechanism to minimize confusion if two people try to manipulate the exact same control. We have opted for simultaneous access because we believe it will encourage each student to explore and control the simulation. Anyone is allowed to do anything at any time. The key to making this work is to provide rich dynamic feedback between students that leaves them constantly aware of each other's actions (Greenberg & Marwood, 1994), and encourages them to talk.

In summary, students have mostly the same image of the core controls, with advanced controls being optional to avoid screen clutter. Anyone can manipulate any control at any time, and all the user's manipulations are constantly visible.

Deictic references allow people to point to things and refer to them using words such as "there", "this one" and "that" (Tatar *et al*., 1991). A strict WYSIWIS view by itself does not provide enough information to let students understand each other's deictic references, for they cannot tell what part of the screen they are attending. Breakdown of deixis has been a common failing of groupware (Tatar *et al*., 1991).

The easiest way to support deictic reference is through *telepointers* (Greenberg, Roseman, Webster & Bohnet, 1992; Tang, 1991), which are cursors, one for each student, that are continuously visible on all displays (as in Figure 3). Telepointers are useful in microworlds for deictic and other types of references. First, they act as a locus of attention; one student can assume that the other is directing their gaze at their cursor. Second, they become an artifact that they can talk around, e.g. the phrase "look at this" is tied to the spot on the screen that the person is pointing to. Third, their animation becomes a gesture. For example, a student circling an area of the screen tells others to attend to all of the items in that area. Finally, they provide a cue of someone's intent. If the telepointer is moving towards a slider, then one expects that the next action could be to change the setting of the slider. This helps mediate who is doing what on the display (Greenberg & Marwood, 1994).

Telepointers were included in most parts of TurboTurtle. People can gesture around the shared view, focus attention to settings on the control panels and implicitly indicate both their intent and their action when manipulating a control.

5. Preliminary evaluation

At all stages of its development, TurboTurtle has been subject to informal usability analysis by HCI professionals and by University students. Only the most recent collaborative version, TurboTurtle-3, has been evaluated on children, and our observations are described in this section.

The aim of the evaluation was to detect user interface flaws and to investigate how children, rather than adults, collaborate in synchronous groupware. Many studies have examined the collaboration styles used by adults when sharing an artifact such as a sketchpad (Tatar *et al.*, 1991; Minneman & Bly, 1991; Ishii, Kobayashi & Grudin, 1992; Greenberg, Hayne & Rada, 1995). We wanted to see whether children's collaboration styles were similar to those of adults, and to see whether they encountered any unforseen problems when interacting through TurboTurtle as a shared artifact. Issues of the pedagogical value of the Newtonian microworld were not investigated—until the coarse grained usability and collaborative issues are addressed and understood they will continually confound any investigation of the subtle pedagogical issues. For similar reasons, at this early stages of usability analysis, we did not investigate the effectiveness of the built-in scaffolding facilities.

Section 5.1 describes the evaluation method. Section 5.2 identifies flaws in the user interface that are independent of the collaborative properties of the microworld. Section 5.3 describes the interaction styles used by the students and discusses their problems with sharing the microworld.

5.1. METHOD

The evaluation was based on a combination of think-aloud (Lewis, 1982) and constructive interaction (O'Malley, Draper & Riley, 1984; Miyake, 1982) evaluation techniques. It was a preliminary study on the general effectiveness of the microworld, and was designed to give quick access to high-level issues such as students' collaboration styles. We felt that more formal techniques, such as controlled experiments, video transcription and so on, would provide too much low-level detail too early.

Twelve students, all aged 10 or 11 years old, used the system in mixed sex pairs for a single half-hour session. We used mixed sex pairs because we wanted to make preliminary observations on whether boys dominate the collaboration. Earlier work on the sharing of a single-user interface in mixed sex young students pairs observed that boys tended to dominate the single set of input and output devices (Bensemann, 1993; Inkpen, Booth, Klawe and Upitis, 1995), although a later study suggests that this may not really indicate who is in control of the group's interaction (Cole, 1995). Additionally, we believed that the use of mixed gender pairings would provide a testing situation rich in potential for collaborative breakdowns—Yelland (1995) reports that mixed gender pairings tend to focus on disagreements rather than clarifications.

The students were seated on swivel-chairs approximately 2 m apart with a clear view of their partner and their partner's screen (pair 1 in Fig. 4). There were several reasons for using a face-to-face evaluation environment. First, we wanted to see whether collaboration through the shared artifact was feasible for young students. If the students chose to get up and walk to their partners' machine this would be a strong indication of the failure of the shared environment: such behaviour would be impossible if students were physically separated. Second, we wanted to see if the students would use eye-to-eye contact as a method of resolving conflicts and breakdowns. Third, if the students chose to watch their partner's screen rather than their own it might indicate a failure of engagement at the interface, or a lack of faith in WYSIWIS.

To promote think-aloud and constructive interaction, all pairs were advised that they should continuously tell their partner what they were doing and why. The components of the system and specific tasks with it were verbally presented to the subjects.

Table 1 shows the order in which system components were introduced to the students. It also shows the higher-level issues that were being implicitly introduced at each stage. The specific tasks that the students carried out varied between subject pairs, but the core of the evaluation was constant, as shown in Table 1. The tasks began by familiarizing students with TurboTurtle as a collaborative tool. For example, initially they were asked to directly manipulate the turtle, with the intended effect that they learn, through their experiences, concepts such as telepointers, collaborative manipulation, and how TurboTurtle can behave as a shared sketchpad. Tasks became increasingly complex, ending

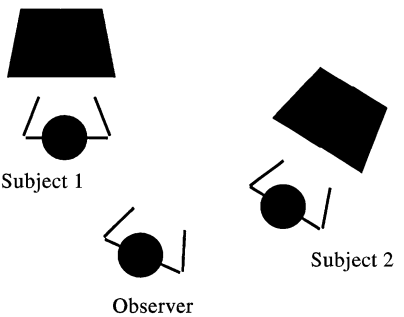

FIGURE 4. Layout of the evaluation environment.

TABLE 1
Order of introduction of system components, and the implicit objective of doing so

Feature introduced	Implicit topic
Direct manipulation of the turtle	With the time off and trail on the sytem behaves as a shared sketch-pad. Telepointers and collaborative manipulation are introduced
Manipulation of turtle size and shape	Introducing sliders as the main mechanism for controlling the microworld. Negotiation over slider control introduced
Trail control: on, off, clearing	Introducing menu selection and absence of WYSIWIS on pulldown menus
System clock, speed, direction, wall-types	Introducing dynamic behaviour of the simulation
X and *Y* component view of speed	Previously Speed had been shown as a vector. Demonstrates a relaxation of WYSIWIS: different views possible
Friction, gravity, mass	Newtonian properties of the turtle
Rocket	Further Newtonian properties of the turtle

with the introduction of Newtonian properties such as friction, gravity, mass, the effects of rockets, and so on. The teacher's module for setting prescriptive tasks (Section 3.4) was not used in this preliminary study. Finally, it should be emphasized that subjects were frequently prompted to think-aloud and to predict system behaviour.

5.2. PROBLEMS WITH THE INTERFACE

Before discussing styles of collaboration and collaborative breakdowns it is necessary to place these issues in context by examining the user's interaction with the system independent of the collaboration. An excess of problems with the single-user aspects of the interface would preclude successful collaboration through it.

Although children were mostly computer literate, they were unfamiliar with the particular widget set used by TurboTurtle. For example, almost all of the children initially had difficulty setting precise values using Tk/Tcl's scale widgets (sliders). This problem was eased when the subjects were shown that clicking to the side of the scale grab-bar, rather than on it, achieved discrete increments of the scale's value.

All student pairs encountered problems with the rocket controls. Most TurboTurtle controls are dynamic—changing a value results in immediate changes in the microworld. The rocket controls are the only exception, with values remaining static for the burn time of the rocket. The problem is severe because the sliders remain active, allowing students to change the *visible* values of the sliders while, counter intuitively, the internal values remain unchanged. This "obvious" user interface flaw was simple to fix, and all rocket controls are now dynamic.

There were several other minor problems with the single-user's interface, but nothing so severe that is precluded successful collaboration.

5.3. STYLES OF COLLABORATION AND COLLABORATIVE BREAKDOWNS

We were interested in seeing how children would collaborate through the microworld. In particular, we wanted to see how they managed (or mis-managed) their interactions in a microworld that not only allowed parallel activity, but that made no attempt to structure turn-taking or mediate conflicting actions.

5.3.1. Observations

Table 2 summarizes the styles of collaboration used by six pairs of users. It shows that different pairs talked to each other in quite different ways (from almost no speech to continuous speech), and that they had various collaboration styles in the ways they jointly manipulated the microworld (from sequential turn-taking to negotiated actions to parallel activity). The table generalizes collaboration styles which evolved during the sessions. For instance, three of the six pairs behaved chaotically in the first moments of the session, changing properties of the simulation rapidly without considering their partners. Two of these three pairs rapidly amended their collaboration styles, without prompting, to allow more considered use of the microworld.

How do students attempt to control, or negotiate control of the system? Our observations show a wide spectrum of styles. Four of the six pairs made some effective use of their ability to simultaneously control the microworld. In particular, pairs two and five

TABLE 2
Summary of the styles of collaboration across the six pairs of users

Pair	Person	Computer familiar	Individual speech	Pair observation
1	Boy	Very	Rapid speech and manipulation	The boy dominated the collaboration with continuous fast speech and rapid manipulation of the microworld
	Girl	Little	Little speech or manipulation	The girl almost totally excluded except when invited to do something by the boy
2	Boy	Very	Continuous discussion	Fluid and dynamic shared control of the microworld
	Girl	Very	Continuous discussion	Periodic breakdowns with appropriate admonishment "You aborted the mission!"
3	Boy	Very	Conversation after breakdown	Extensive negotiation over the management and ordering of activities with the girl taking the leading role
	Girl	Very	Continuous "think aloud"	
4	Boy	Very	Continuous speech	The boy primarily drove the collaboration, with continuous invitations for the girl to carry out activities
	Girl	Moderate	Continuous response	
5	Boy	Very	Continuous discussion	Good shared control of the microworld
	Girl	Very	Continuous discussion	
6	Boy	Very	Almost no speech	Very poor use of the microworld, each user attempting to make single user use
	Girl	Little	Almost no speech	

continually discussed their actions and managed their collaborations successfully. Interestingly, these successful pairs were vocal about the undesired actions of their partners. For instance, in pair two, the boy closed off the rocket control dialogue (by clicking the "Abort Mission" button) without prior warning. The girl immediately turned to the boy and scolded him with "You aborted the mission!"

Other pairs demonstrate extremes of collaborative breakdown. Pair one, for instance, demonstrated the problems that arise when one person dominates the collaboration. The boy changed the properties of the simulation so rapidly that the girl could not keep pace. The girl initially took her hands away from the mouse, clearly attempting to follow the frenetic activity of her partner. Shortly afterwards she shouted "Leave it!" The boy briefly capitulated. For instance, when about to delete the turtle's trail, he asked "Do you want that to stay there?" However, he continued to dominate the session, grabbing the turtle or sliders whenever the girl hesitated. This behaviour is consistent with the observations of Bensemann (1993) who notes that strong-willed and highly independent children failed to collaborate successfully, and that boys tend to dominate the control of input devices when sharing a single-user system.

In contrast, pair three followed sequential activity, in which they negotiated control to the almost total exclusion of simultaneous activity. For example, when asked to set the rocket heading to zero, its fuel to 50, and its force to 100, the girl said "You set the heading, then I'll set the fuel and force. Then you can launch the rocket". The sequence of actions was carried out in that order, with no overlapping of actions, and with an explicit "OK" once each stage was completed.

Pair six almost ignored the fact that they were in a collaborative microworld. They were almost silent despite being frequently encouraged to speak to each other. They struggled against the actions of each other, even though the telepointers showed the cause of their difficulty (the evaluator confirmed that they were aware of the purpose of the telepointers). At the end of their session they stated that the microworld would be much easier to use on their own. Their reticence during the evaluation cannot be attributed to shyness as they were open and articulate when interviewed at the end of the session.

5.3.2. Conflicts and breakdowns

The "successful" collaborations, indicated by pairs 2–5, were far from seamless, with breakdowns frequently occurring. Many of the collaborative "breakdowns", however, were positive contributions to the overall interaction, with the breakdown becoming a focal point for negotiation about what the microworld should do next. Conflicts over the sliders and turtle were frequent, especially if a task consisted of a single unit task (such as "Move the turtle to the top-right of the screen", or "Make the Mass 5"). The result of such conflicts is the TurboTurtle equivalent of Window-Wars (Stefik, Foster, Kahn, Bobrow, Lanning & Suchman, 1988) where the turtle or slider position jerks in response to simultaneous manipulation. However, such conflicts are not necessarily a bad thing. Greenberg and Marwood (1994) argue that simultaneous access to controls can be mediated by showing people that conflict is occurring, and that the participants can repair the conflict through their natural social skills, much as they do in the real world. In TurboTurtle, the children could see the two telepointers on the slider as well as the bouncing slider position as both tried to move it. The resulting problem was more in

their own immaturity at negotiating control. For example, in some cases the subjects were tenacious in their desire to be last one in control, even though they were well aware of the cause of the problem as revealed by the telepointer.

What distinguished a successful "breakdown" from an unsuccessful one was the extent of discussion that accompanied the conflict. Pair two, for instance, normally managed the sliders without difficulty, but at one point they argued over the desired mass of turtle: the girl trying to set the mass to 20, while the boy tried to make it 30. Their conflict lasted approximately 8 seconds and was accompanied by continuous comments such as "Make it 20!", "No! Make it 30!" and so on. Finally the girl set the mass to 20, and the boy commented "Well I'll make it bigger then". What is important here is that the conflict stems from the task, rather than the interface. In contrast, pair six encountered the same problem of simultaneous access to a slider, but it was not clear to the observer (or to the collaborators) whether the values that they were trying to set were the same or different—the confusion being caused by a total absence of speech.

5.3.3. "Bugs" in the support for collaboration

The importance of mutual activity awareness in groupware was emphasised by a flaw in the rocket controls. As described earlier, TurboTurtle's "telepointers" allow each user to maintain awareness of the activity of their colleagues by communicating each user's cursor position. GroupKit, the underlying groupware toolkit (Roseman & Greenberg, 1996), is limited to a single set of primitive telepointers within an application's top-level main window. Telepointers were unavailable when changing rocket properties because the rocket controls were implemented in a separate top-level window.

The absence of awareness information when working with the rocket was a significant problem for all the students. Frustrated comments such as "Hey, how did that happen", and "What are you doing?" were frequent.† Clearly, the absence of telepointers in this part of the dialog and the associated breakdowns indicates how important it is to maintain the children's awareness of each other's activities.

Other "bugs" in the collaborative support are harder to pin-point. Our general impression on WYSIWIS was that successful collaboration pairs wanted a more rigid implementation of WYSIWIS: for instance pairs four and five wanted to be able to see their partner's menu selections (pull-down menus were only shown to the user executing the action), and they wanted to maintain a shared view of the speed controls. Pairs one, three and five said that the telepointers should be bigger or brighter to help maintain awareness, and pair six stated that they found it hard to see the actions of their partner, but that the cursors should not be made bigger or brighter. All these comments clamour for more awareness of each other's actions in all parts of the microworld (Gutwin, Stark & Greenberg, 1995).

5.4. SUMMARY OF THE EVALUATION

The children had fun. All students indicated that they enjoyed using the system, and advised that more colours, sounds and better graphics would improve the system. None

† To provide telepointers in a separate rocket window would require additional functionality at the toolkit level. Until this is provided, telepointers will be enabled by attaching rocket controls to the main TurboTurtle window.

of the students chose to leave their machine and work directly with their partner on a single machine. Eye-to-eye contact was very rare, but during breakdowns it was common for one subject to glance at their partner, without reciprocation, and then return their gaze to their own screen. After an initial confirmation of the system's WYSIWIS behaviour, by glancing at their partner's screen at the start of the session, each user's gaze did not return to their partner's screen. These observations indicate a generally successful implementation of WYSIWIS.

We see the evaluation as encouraging. We believe the interaction styles witnessed are somewhat comparable to those previously reported for collaboration between adults. However, children's negotiation of control are almost caricatures of adult negotiation. Even in this small subject set, we saw extremes in the way individual children's personalities affected the successes of their partnerships. Although we have not been surprised by the breakdowns that occurred, we were surprised by the extreme behaviours we observed, and by the persistence of some children to continue actions that limit the effectiveness of the partnership. In retrospect, we should have predicted this outcome given the youth of our users and the (sometimes) oil and water relationships caused by boy/girl partnerships (Yelland, 1995).

The TurboTurtle study also supports Cole's argument that students' control of a collaborative activity is a social process developed by group dynamics, rather than an artifact induced by mouse possession (Cole, 1995). Cole noticed that the single mouse can serve both as a symbol of subordination when the mouse-holder is obeying group instructions, or as a symbol of control when the mouse-holder makes the decisions. In TurboTurtle, which provides multiple mice and parallel activity, the issue of control was clearly a social one. Even though mouse possession played no role, we still saw a flux where students assumed subordinate positions, fought for control or dominated the collaboration.

In summary, our observations support the argument that groupware cannot make a bad team good, but we have been sufficiently encouraged to continue development and evaluation. We believe TurboTurtle to be an iteration towards a collaborative and educational virtual laboratory. We have to consider how collaborative systems can both give children the freedom to explore the world at their own pace and personal style, while adding structure to minimise the risk of breakdowns expected because of the immaturity of the audience.

6. Related work

Single-user microworlds have been developed since Papert envisaged them in Mindstorms (1980). Smith's Alternative Reality Kit (1987) is one example, and was a direct stimulus for our initial work on TurboTurtle. Our current research interests, however, are on the design and use of synchronous collaborative artifacts by children. The works most closely related to these interests stem from the Envisioning Machine (Roschelle, 1991; Roschelle & Clancey, 1992) and from SharedARK (Smith, O'Shea, O'Malley, Scanlon & Taylor, 1991).

The Envisioning Machine is a single-user system for examining acceleration and velocity. Despite being a single-user system, Roschelle and Clancey (1992) examine collaborative learning with pairs of students sharing the Envisioning Machine's single set

of input and output devices. Their work emphasises the user's formation of a mutual understanding through simultaneous coordination of the three following activities: perception of the shared artifact, gesture around it, and language. They report that, in order to understand the students' interactions, it is necessary to analyse all three communicative channels. Our observations with TurboTurtle concur that all three channels are fluidly mixed in effective collaboration, and that the breakdown of one channel corrupts the overall interaction. For instance, pair one's failure to talk destroyed their collaboration (a language failure), and the absence of telepointers in the rocket dialogue severely damaged its effectiveness in support for collaboration (a gestural and artifact failure). It is notable that, in contrast to the Envisioning Machine, gestures around TurboTurtle are *part* of the artifact (telepointers) rather than being an activity external to it.

Like TurboTurtle, SharedARK is a synchronous groupware application that simulates a variety of physical objects. The primary question motivating the research reported in Smith *et al.* (1991) is "What is different when members of a problem-solving pair are physically separated then reconnected via [different types of] computer and communications technology?" Their study, then, focuses on comparing physically remote collaboration supported by video-tunnels, with physically remote audio-only collaboration, with face-to-face collaboration in a variety of different seating positions. The subjects in the study were mainly postgraduate research students with some technical and managerial employees of a University. To our knowledge, SharedARK has not been used to examine collaboration between young students.

Other related work, reported in CSCL'95 (Schnase & Cunnius, 1995), reveals a variety of techniques for supporting and examining collaborative work. For instance, Bricker, Tanimoto, Rothenberg, Hutama & Wong (1995) examine collaboration between eighth grade math students and between college students using a suite of small shared artifacts such as the *The Color Matcher*. In the *Color Matcher* users adjust red, green and blue colour values to match a 'target' colour. Although all of the systems in the suite provide multiple input devices (each student has a mouse to control a private cursor), they only support a single output device (a shared screen). Furthermore, there are severe constraints on the GUI components that each user can control. Each input wilget, such as the slider controlling the "Red" value, can only be manipulated by the user with the red cursor. They report that allowing multiple input devices (mice) made for "very little conflict between the users as there was no contention for the input resources", but that the restriction on simultaneous control caused problems—"some students did get slightly frustrated in discovering that they could not manipulate another player's color". Also reported in CSCL'95 are two analyses of mixed gender pairings when sharing single-user software (Inkpen *et al*., 1995; Yelland, 1995). Inkpen *et al.* note the problems that young students have when sharing a single mouse, and stress the potential of systems that support close collaborative work as well as individual exploration.

7. Technical design experiences

This section briefly examines the implementation efforts of the three TurboTurtle iterations. Our most significant observation is that building computer-supported cooperative learning (CSCL) systems is no longer solely the domain of computer hackers

and technical gurus. The availability of graphical and groupware toolkits alleviate much of the complexity.

The command-line TurboTurtle-1 was implemented in approximately 5000 lines of C on a VAX 11/780 using a BBC Model B micro-computer to display the graphical output. Its hardware demands made it highly inappropriate for general classroom use, but it was a valuable point system for our further work.

The graphical TurboTurtle-2 was written in approximately 1000 lines of Tcl/Tk code (Ousterhout, 1993) on a Unix environment. That its implementation is only one-fifth the size of the original version, in spite of the much higher complexity, is indicative of the state of the art in graphical user-interface (GUI) toolkits. In our particular case, the interpreted scripting language of Tcl made it easy to describe TurboTurtle's flow of control, and the widget sets provided by Tk dramatically simplified the actual construction of the controls and the simulation.

The collaboration-aware TurboTurtle-3 was built in GroupKit, a groupware toolkit (Roseman & Greenberg, 1992; Roseman & Greenberg, 1996). Because GroupKit is an extension to Tcl/Tk, we directly modified TurboTurtle-2 to make it group-aware. Minor modifications were made to about 80 lines, and only about 50 new lines of the code were added to the original 1000. This took about 8 hours to do. As a result, TurboTurtle gained its extensive facilities for group-awareness, such as telepointers and WYSIWIS display. It also acquired the ability to bring latecomer to a session up to date, ensuring that their view of the microworld is same as their fellow students.

The fact that minor changes and very little effort was required to make TurboTurtle group-aware is a direct result of GroupKit. Since this toolkit foreshadows the kinds of groupware tools that CSCL developers will have at their disposal, it is worth summarising GroupKit's features.

1. A *runtime infrastructure* automatically creates the necessary distributed processes on all machines, and manages their interconnection and communication requirements.
2. A simple set of *groupware programming abstractions* gives the developer most of the tools required to coordinate their groupware applications. Primitives include remote procedure invocation between application instances, sharing of data, and generation and tracking of conference events (such as the arrival and departure of participants).
3. A set of *groupware widgets* lets developers easily add generic interface constructs of value to conference participants. These include telepointers, participant widgets that display who is in a conference and provide information about them, and widgets that promote awareness of where others are working in a relaxed WYSIWIS view.
4. *Session managers*, interfaces that let people connect to groupware conferences, are separated from the groupware applications. This means that the application developer can concentrate on the application itself, rather than on how people connected to each other.

Most of the changes to TurboTurtle-2 simply required us to use GroupKit's remote procedure call facility to tell all processes to execute an action at all sites. Examples include telling all to move the turtle to a new position, or updating the display on a slider.

The telepointers were added with two lines of code, as was the widget listing the students using the microworld. There were a few special things that had to be done, but none were onerous. Perhaps the best part was that many technically complex aspects of groupware could be ignored, simply because they were handled automatically by GroupKit (examples include session management and communication setup).

While TurboTurtle-3 was easy to code, we are not implying that adding group-awareness is a trivial matter. The point is that the simplicity of GroupKit allowed us to concentrate more on the design of TurboTurtle-3, rather than its coding.

8. Summary

TurboTurtle has gone through several design iterations, starting from a command-line system and ending as a collaborative simulation environment. The paper described its design evolution, and listed several of the design rationales behind our decisions.

There are many directions for further work in TurboTurtle. With respect to refinements of the microworld, the world's our oyster: there is no obvious end to the types of domain that can be covered by a group-aware simulation. There is, of course, much to be done investigating the nuances of adding collaboration to learning environments. Although we have run several *ad hoc* usability studies and one larger collaborative usability study we believe that, as yet, we are only detecting the "large grain" usability flaws. Extensive observation over the coming years will refine our understanding of the nature of computer-supported collaborative learning.

Our observations and experiences in developing a collaborative microworld should be of interest across the various disciplines in computer supported learning. Of note is the small effort required to convert our single-user microworld to a collaborative one. Toolkits for real-time groupware, such as GroupKit, are greatly reducing the prerequisites of technical knowledge demanded to build a new generation of multi-user educational systems.

The collaboration between the two was made possible by the University of Canterbury's Erskine Fellowship. The GroupKit part of this research is (gratefully) supported in part by the National Engineering and Research Council of Canada, and by Intel Corporation. Many people at the University of Calgary, with Mark Roseman in particular, have contributed in one way or another to GroupKit's development.

References

BAECKER, R. M. (1993). *Readings in Groupware and Computer-Supported Cooperative Work*. Los Altos, CA: Morgan Kaufmann.

BENSEMANN, G. (1993). *Capturing the Interest of Young Children in Computer Science*. B.Sc. (Honours) Project, Department of Computer Science, University of Canterbury, Christchurch, New Zealand.

BRICKER, L. J., TANIMOTO, S. L., ROTHENBERG, A. I., HUTAMA, D. C. & WONG, T. W. (1995). Multiplayer activities that develop mathematical coordination. *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, pp. 32–39. Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.

CLEMENTS, D. H. & GULLO, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, **76,** 1051–1058.

COCKBURN, A. & GREENBERG, S. (1995). Turbo-Turtle: a collaborative microworld for exploring Newtonian physics. *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, pp. 62–66. Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.

COCKBURN, A. J. G. (1994). Turbo-Turtle: educating children in an alternative reality universe. *Proceedings of the Computer Human Interaction Specialist Interest Group of the Ergonomics Society of Australia (OZCHI'94)*. 28 November–1 December, pp. 99–105, Melbourne.

COLE, K. A. (1995). Equality issues on computer-based collaboration: looking beyond surface indicators. *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, pp. 67–74. Bloomington, IN, 17–20 October, London: Lawrence Erlbaum Associates, Inc.

GREENBERG, S. & MARWOOD, D. (1994). Real time groupware as a distributed system: concurrency control and its effect on the interface. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 22–26 October, pp. 207–217. Chapel Hill, North Carolina.

GREENBERG, S., ROSEMAN, M., WEBSTER, D. & BOHNET, R. (1992). Issues and experiences designing and implementing two group drawing tools. *Proceedings of the 25th Annual Hawaii International Conference on the System Sciences*, Vol. 4, pp. 139–150. Hawaii, January.

GREENBERG, S., HAYNE, S. & RADA, R. (1995). *Groupware for Real Time Drawing: A Designer's Guide*. New York: McGraw-Hill.

GRUBER, H. E. & VONECHE, J. J. Eds. (1987). *The Essential Piaget: An Interpretive Reference and Guide*. New York: Basic Books.

GUTWIN, C., STARK, G. & GREENBERG, S. (1995). Support for workspace awareness in educational groupware. *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, pp. 147–156. Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.

HADFIELD, O. D., MADDUX, C. D. & HART, C. (1989). Effectiveness of Logo instruction in reducing mathematics anxiety among eight-grade students. *Computers in Schools*, **6,** 103–112.

HILL, R. D., BRINCK, T., PATTERSON, J. F., ROHALL, S. L. & WILNER, W. T. (1993). The rendezvous language and architecture. *Communications of the ACM*, **36,** 62–67.

INKPEN, K., BOOTH, K. S., KLAWE, M. & UPITIS, R. (1995). Playing together beats playing apart, especially for girls. *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, pp. 177–181. Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.

ISHII, H., KOBAYASHI, M. & GRUDIN, J. (1992). Integration of inter-personal space and shared workspace: ClearBoard design and experiments. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 33–42. Toronto, Canada, 31 October–4 November.

KLAWE, M. & PHILLIPS, E. (1995). A classroom study: electronic games engage children as researchers. *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, pp. 209–213. Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.

LEWIS, C. (1982). *Using the 'thinking-aloud' method in cognitive interface design*. Technical Report RC9265, IBM T.J. Watson Research Center, Yorktown Heights, NY.

MADDUX, C. (1985). The need for science versus passion in educational computing. *Computers in Schools*, **2,** 9–10.

MADDUX, C. D. & LAMONT-JOHNSON, D. (1988). *Methods and Curriculum for Teachers*. Hawthorn Press.

MANTEI, M. (1988). Capturing the capture lab concepts: a case study of in the design of computer supported meeting environments. *Proceedings of the 2nd Conference on Computer Supported Cooperative Work*, pp. 257–270. Portland, Oregon, 26–28 September.

MINNEMAN, S. L. & BLY, S. A. (1991). Managing a Trois: a study of a multi-user drawing tool in distributed design work. *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*, pp. 217–223. New Orleans, May.

MIYAKE, N. (1982). *Constructive interaction*. Technical Report, CHIP Report 113, Center for Human Information Processing, University of California at San Diego.

O'MALLEY, C., DRAPER, S. & RILEY, M. (1984). Constructive interaction: a method for studying human–computer–human interaction. *Interact '84. The 1st International Conference on Human–Computer Interaction*, pp. 269–274. London, UK, 4–7 September.

OUSTERHOUT, J. K. (1993). *An Introduction to Tcl and Tk*. Reading, MA: Addison-Wesley.
PAPERT, S. (1980). *Mindstorms—Children, Computers, and Powerful Ideas*. Harvester Press.
PAPERT, S. (1993). *The Children's Machine: Rethinking School in the Age of Computer*. New York: Basic Books.
PRESS, L. (1996). Seeding networks: The federal role. *Communications of the ACM*, **39,** 11–18.
ROSCHELLE, J. (1991). *Students' Construction of qualitative physics knowledge: learning about velocity and acceleration in a computer microworld*. Ph.D. Thesis, University of California, Berkeley.
ROSCHELLE, J. & CLANCEY, W. J. (1992). Learning as social and neural. *Educational Psychologist*, **27,** 435–453.
ROSEMAN, M. & GREENBERG, S. (1992). GroupKit: a Groupware Toolkit for building real-time conferencing applications. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 43–50. 31 October–4 November, Toronto, Canada.
ROSEMAN, M. & GREENBERG, S. (1996). Building real time Groupware with GroupKit, a Groupware Toolkit. *ACM Transactions on Computer–Human Interaction* (in press).
ROSSEN, M. B. & CARROLL, J. M. (1996). Scaffold examples for learning object-oriented design. *Communications of the ACM*, **39,** 46–47.
SCHNASE, J. L. & CUNNIUS, E. I., Eds. (1995). *ACM Conference on Computer Supported Cooperative Learning (CSCL '95)*, Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.
SHNEIDERMAN, B. (1987). Direct manipulation: a step beyond programming languages (excerpt). In R. M. BAECKER & W. A. S. BUXTON, ed. *Readings in Human–Computer Interaction: A Multidisciplinary Approach*. Los Altos, CA: Morgan Kaufmann.
SMITH, D. C., CYPHER, A. & SPOHRER, J. (1994). KIDSIM: programming agents without a programming language. *Communications of the ACM*, **37,** 55–67.
SMITH, R. B. (1987). Experiences with the alternate reality kit: an example of the tension between literalism and magic. *Proceedings of ACM CHI + GI'87 Conference on Human Factors in Computing Systems and Graphics Interface*, pp. 61–67.
SMITH, R. B., O'SHEA, T., O'MALLEY, C., SCANLON, E. & TAYLOR, J. (1991). Preliminary experiments with a distributed, multi-media, problem solving environment. sharedARK. In J. M. BOWERS & S. D. BENFORD, Eds. *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*, Amsterdam: North-Holland.
SOLOWAY, E. & PRYOR, A. (1996). The next generation in human–computer interaction. *Communications of the ACM*, **39,** 16–18.
STEFIK, M., BOBROW, D. G., FOSTER, G., LANNING, S. & TATAR, D. (1987). WYSIWIS revised: early experiences with multiuser interfaces. *ACM Transactions on Office Information Systems*, **5,** 147–167.
STEFIK, M., FOSTER, G., KAHN, K., BOBROW, D. G., LANNING, S. & SUCHMAN, L. (1988). Beyond the Chalkboard: computer support for collaboration and problem solving in meetings. In I. GREIF, Ed. *Computer Supported Cooperative Work: A Book of Readings*. Los Altos, CA: Morgan Kaufmann.
TANG, J. C. (1991). Findings from observational studies of collaborative work. *International Journal of Man–Machine Studies*, **34,** 143–160.
TATAR, D. G., FOSTER, G. & BOBROW, D. G. (1991). Design for conversation: lessons from Cognoter. *International Journal of Man–Machine Studies*, **34,** 185–209.
THIMBLEBY, H. (1990). *User Interface Design*. New York, Reading, MA: ACM Press, Addison-Wesley.
VAIDYA, S. & MCKEEBY, J. (1985). Conceptual problems encountered by children while learning Logo. *Journal of Educational Technology Systems*, **13,** 33–39.
YELLAND, N. (1995). Collaboration and learning with Logo: does gender make a difference? *ACM Conference on Computer Suported Cooperative Learning (CSCL '95)*, pp. 397–401: Bloomington, IN, 17–20 October. London: Lawrence Erlbaum Associates, Inc.

Paper accepted for publication by the Editor, Dr. B. Gaines.

Availability

A world wide web page describing GroupKit, and providing access to it via anonymous ftp, is in `http://www.cpsc.ucalgary.ca/grouplab/groupkit`
While GroupKit is not without its limitations, it is an excellent tool for CSCL researchers who wish to prototype groupware systems and examine how they are used.

TurboTurtle is available directly from the first author of this paper.